A Market Based Approach for Resolving Resource Constrained Task Allocation Problems in a Software Development Process

Murat Yilmaz[1,2] and Rory V. O'Connor[2,3]

[1] Lero Graduate School in Software Engineering, Dublin City University, Ireland
murat.yilmaz@computing.dcu.ie
[2] Dublin City University, Ireland
[3] Lero, the Irish Software Engineering Research Centre
roconnor@computing.dcu.ie

Abstract. We consider software development as an economic activity, where goods and services can be modeled as a resource constrained task allocation problem. This paper introduces a market based mechanism to overcome task allocation issues in a software development process. It proposes a mechanism with a prescribed set of rules, where valuation is based on the behaviors of stakeholders such as biding for a task. A bid process ensures that a stakeholder, who values the resource most, will have it allocated for a limited number of times. To observe the bidders behaviors, we initiate an approach incorporated with a process simulation model. Our preliminary results support the idea that our model is useful for optimizing the value based task allocations, creating a market value for the project assets, and for achieving proper allocation of project resources specifically on large scale software projects.

Keywords: Software Process Improvement, Game Theory, Process Simulation, Mechanism Design, Auction Mechanism, Task Allocations.

1 Introduction

Software development is an organized social setting, which should be equipped with economic methods for producing products in a multi-stakeholder viewpoint. While coping with uncertainties, activities of software development place precious *resources at risk* [1]. Conceptually, software development is also a form of economic activity, whereas its organizational structure should be considered as a social (decision-making) system based on several networks of interactions [2]. In this particular perspective, complexity of software development does stem from the complexity of human interactions and social communication costs [3], and therefore can be investigated as an organizational design problem.

Several empirical observations suggests that many different software projects not only fail due to technical reasons but also fall through organizational or team incompatibilities, and recently there is much interest in the social impacts of a

software development process [4]. Furthermore, any intellectual process like software development should take into account that the knowledge used in software practices is tacit, dynamic and most importantly embedded in social relations [5].

Stellman [6] reported that productive team formation is a very vital component of management process. Most importantly however, the challenge here is to constitute a methodology by valuing resources with a decentralized *modus operandi*, which projects the burden of task planing onto the individuals responsible for carrying out specific tasks. To deal with self interested participants who can selfishly consume resources, the concept of mechanism design (MD) - a field of economic theory - has been found useful among community of researchers. For example, it founds an application in the field of computer science as *algorithmic game theory* [7]. While social choice theory claims that it is possible to merge participants' preferences into a single utility (i.e. preference) function, the goal of MD is to optimize these social choices based on the accumulation of individuals' preferences. MD constitutes a collective decision-making process with the assumption that participants will act rationally as defined in game theory. As a software project expands in its strategic settings, it becomes more convenient for management to induce collective decisions as a social choice function to reduce the decision to a single alternative, where several tasks are owned and operated by different parts of the software development organization.

The objective of this paper is to establish a novel approach for analyzing development task-resource allocation problems in the software development process by using a market based mechanism design approach. Our aim is to optimize the task-resource allocations based on the bids of the participants and decentralized market rules. The problem discussed here is constructed in two dimensions; firstly, as a theoretical model, which includes resource allocation rules and their symbolic representations. Secondly, by using simulated pseudo data for an initial test of our model, we develop a process simulation by exploiting kanban as a software development process.

The remainder of the paper is organized as follows. In section 2, we review the literature relating to the use of game theory in the area of software engineering. We identify several cases where MD is used to resolve resource allocation problems in information systems management. In section 3, we propose an auction-based market mechanism, which is constructed for addressing resource allocation issues in software project management. In section 4, we illustrate our model by using a virtual software project. Finally, in section 5, we draw our conclusions with respect to our implementation of the suggested market mechanism.

2 Game Theory in Software Engineering Literature

Several limited attempts have been made to understand software development as a cooperative or a competitive game form. For example, Lagesse [8] build a model based on a cooperative game theory approach with the idea of optimizing task assignment in software engineering efforts. On the other hand, Grechanik and Perry [9] focus on a game theoretic approach as a non-cooperative game,

based on the fact that there are a number of potential goal conflicts among the roles of a software development approach. Moreover, Cockburn [10] consider software development as a series of games of invention and communication, where he portrayed the software development as "*economic-cooperative gaming*". His vision is similar to an iterative game in which two goals are competing for a resource. He also suggested that as an emerging area, which he called "mechanics and economics of communication" should be investigated in the near future. Based on the skills of the participants, Cockburn [11] also points out that software development should be considered as a game constrained upon its project resources. Using a approach based on grounded theory, Baskerville *et al.* [12] considered trade-offs and balancing decisions as balancing games that may appear in three different levels (i.e market, portfolio, management), where their nature is to progress dynamically with the demands of a market. Ko *et al.* [13] use a game theoretic approach for improving the reliability of data collected by using a method to improve its accuracy for better quantitative process management, where they also recommend a study for applying game theory in software project management and software process improvement activities. To improve the learning abilities of students Holeman [14] design a software process improvement game, which is a type of board game (designed to instruct CMMI to students) that participants compete for achieving CMMI level 2 on a Monopoly-like game board. Ogland [15] develops an approach for conflicting situations by using game theory and drama theory. He portrays software process improvement (SPI) as a game playable by quality auditors, software engineers, and managers. The goal is to identify how an SPI standard progress through an equilibrium (i.e. a proposed solution concept in a game).

Although game theory can be considered as a new and emerging field, there are a variety of related works outlined the importance of decision-making in software development landscapes. Equipped with the idea of "making everyone a winner", Theory W [16] is an approach based on the concept of risk management in software engineering decisions. To resolve the conflicts among the stakeholders of a project, it also suggests that the role of management somehow acts like a mediator or a negotiator, which seems likely similar to a game theoretic approach. In order to establish *a value based approach* and formalize the design goals of software development, Sullivan *et al.* [17] consider software design as an investment activity, where they applied the concept of real options to evaluate economic outcomes. To improve the effectiveness software architecture decision-making, Vajja and Prabhakar [18] investigate design issues based on the quality attributes, where they can be modeled as a game theoretical problem. Sazawal and Sudan [19] suggest a game model named, as a *basic software evaluation game* seems to be useful for helping software teams on decision-making particularly from an evolutionary perspective on software design decisions. Furthermore, they hypothesize that *lightweight* game theory is more useful for understanding software evolution. Bavota *et al.* [20] investigate the opportunities for using non-cooperative game theory for "extract class refactoring" in a situ-

ation such as two players that are competing to build new classes for improving the levels of cohesion.

There are some works in software engineering literature for the application of Prisoner's Dilemma (i.e. a non-cooperative game, based on two persons interactions). For example, Hazzan and Dubinsky [21] investigate the way of cooperation in extreme programming, in particular for pair programming practices. Secondly, a hidden game of Prisoner's Dilemma is investigated by Feijs [22] between a programmer and a tester. Thirdly, Oza [23] uses Prisoner's Dilemma framework to investigate strategic interactions in a client-vendor relationship in offshore outsourcing projects. Recently, Klein *at al.* [24] draws out attention to the notion of incentive conflicts in a software development both for identifying design characteristics and resource allocation perspectives. To bridge the gap between these conflicts of interests, they suggest that the notion of game theory particularly in terms of mechanism design should be useful for improving incentive compatible, decentralized and dynamic decision-makings in the software development processes.

2.1 Mechanism Design

The notion of MD is about understanding the structure of an organization such as a communication system for improving social decision-making and societal welfare. In MD, a social planner can create organizational structure to induce a planned or desired outcome based on the private information hold by the participant's of an organization. The information provided in this process is useful for modeling organizational procedures, solving economic problems such as allocation of resources, or dealing with problems related with asymmetric information and ultimately for supporting cooperation among the organization [25]. MD should also assist a social planner to model an organization for analyzing how the private information of individuals interacting throughout the organizational rules, which directly affect the expected outcomes. Such a model usually depends on the information of what is the possible action for each participant and their consequences that constitute the allocation decisions as a game theoretical solution.

Zhao *et al.* [26] propose an approach for understanding of Internet security issues as economical factors such as factors govern the actions and interdependence of the participants. To this purpose, they implement an economic mechanism (in this context, a certification mechanism) for reducing the security risks of users over the Internet. The essence of this mechanism depending on the idea to minimize the possibility of sending out malevolent traffic by increasing the responsibility of service providers and promoting the incentives to monitor the suppliers of malware and spam in their networks. The mechanism best works on a certified network concept by which each certified service provider will be able to use the collected information from other providers and held responsible for the traffic that is generated by their users [26].

Stef-Praun and Rego [27] outline a simple mechanism to transfer system wide efficient allocations of resources rather than individual resource allocations in a

decentralized market for web services producers and consumers. Authors claim that the proposed mechanism can be realized to fit any structure composed of a large number of self-interested participants (e.g. a dynamic collaborative environment). Friedman and Parkes [28] investigate a customer pricing problem of a wireless networking provider, which may seen in a coffeehouse as a mechanism design problem. They develop a game theoretical model for bandwidth allocation based on a game of incomplete and asymmetric information

In summary, these findings suggest that the mechanism design theory and its actual implementation for software organization can help for analyzing several economic interactions and designing organizations including markets and auction based market designs.

3 An Auction-Based Market Mechanism

A research focus of software process improvement is to allocate the project resources more efficiently, which is crucial for the software project's overall success. Similar to an economy, the process of software development consist of many independent parties (i.e. stakeholders) with autonomous (sometimes conflicting) objectives. These parties also have their private information (e.g. personal preferences), which should be revealed to improve the socioeconomic success of software development.

In an auction based MD, a market designer (e.g. economist, mediator, manager, etc.) is responsible for regulating the interactions of individuals, who promote social or economic objectives, for example; creating the right incentives for improving participants productivity. An example for such a mechanism is an auction where participants are defined as bidders that are bidding for the resources. These bids, however, may not value their requests truthfully. One way to deal such situations is to implement a *Vickrey auction* (i.e. a second price auction), which is based on a rule that each bidder submit a (sealed) bid and the valuation for a price is chosen as the second highest bid that is also paid by the winner. In theory, an aim is to maximize the efficiency of resource allocation by having a proper valuation.

3.1 Our Approach

Here, we formalize a software project as a market-based auction mechanism where the activities of the project is transformed into tasks. By using a role, *mediator*, these tasks are announced by an auction classification procedure. This rule set automates, which resources will be used for how long. Next, the auction system creates the auction and waits for the highest bidder.

Assume that a task is auctioned among n participants $i = 1, ..., n$, where valuation of an item by an individual is v_i for this item. The preference of a participant is given as a valuation function from an action set a, where $v_i : \mathcal{A} \rightarrow \Re$. Suppose that player i bids are in a vector $b = (b_1, b_2, ..., b_n)$.

The function which is used for delivering the task to a winning participant;

$$f_i(b_1, b_2, ..., b_n) = \begin{cases} 1 \text{ if } b_i > b_j, j = 1, ..., i-1, i+1, ..., n \\ 0 \text{ otherwise} \end{cases} \quad (1)$$

The utility function of each bidder can be shown as;

$$u_i(b_1, b_2, ..., b_n) = x_i(b_1, b_2, ..., b_n)(v_i - p_i(b_1, b_2, ..., b_n)) \quad (2)$$

Consequently, the participant who values the item most is the winner from the set of participants $\mathcal{P} = \{i_{wins} | i \in I\}$ with the highest declared value and by following second price (Vickrey) auction, (see Listing 1.1 for a Mathematica routine to perform a second price auction). Please note that if there are tied values, a randomize function will be executed to resolve the issue.

Listing 1.1. Second Price Auction (adapted from [29])

```
SecondPrice[Bids_] :=
Module[
 {ii, Players, TiedValues, Winner, WinnerValue},
 Players = Table[ii, {ii, 1, Length[Bids]}];
 g[i_] := Bids[[i]] == Max[Bids];
 TiedValues = Select[Players, g];
 Winner = TiedValues[[Random[Integer,
     {1, Length[TiedValues]}]
   ]
  ];
 LowerSet = Complement[Players, {Winner}];
 WinnerValue = Max[Bids[[LowerSet]]];
 Return [{Winner, WinnerValue}]
 ]
```

3.2 Auction Basics

To realize the *true* economic value of a task, we propose to decompose every divisible parts of project tasks to form a set of auctions. There are two different roles interacting in this paradigm. Firstly, the people who are able to create auctions are called the *auctioneers*. For example, they can be an individual stakeholder or team of employees, who is authorized to construct a relationship between project resources versus potential software tasks. To reveal their true value, they create task based auctions in the market system. Secondly, the participants, who are interacting with the auction mechanism is called the *bidders* (e.g. software developers, testers, analysts, etc.). By using the auction mechanism, our system treats individuals or teams as entities that compete for allocation of project tasks.

In theory, we assume that bidders, who has best bidding plans seek to maximize the sum of their valuations. Furthermore, one important reason for creating such mechanism enables us to micromanage the idle tasks that are not utilized. However, this ability is not possible in many conventional approaches. Two of the auction features we suggest; (i) do not allow participants to enter a number consecutive bids for the same item and (ii) use a type of credit system similar

to money or other incentives. Ultimately, this means that we enable auctioneers to create auctions on a time frame with the credits they can spend on auctioned tasks, which is scalable by allocating a suitable budget for required tasks and performance estimations (see Figure 1).

3.3 Rules to the Auction

- Our auction design aims to result in multiple rounds of concurrent bids for each task defined by the market.
- Auctioneer creates an auction, where initially the proposer should have determined the value proposition for every task. However, this value shall fluctuate with respect to the market requirements.
- In the first round, each bidder in the auction system makes a bid on one task that is auctioned. To remain in the auction bidders should keep their status active on the system (e.g. next round in auction).
- A bidder defined by the auction system shall be bidding on at least one task.
- An active bidder either currently holds the top bid on a particular task, or else raises the bid on a task of the bidder's choice by at least the minimum bid increment.
- A bidder who is in the possession of the top bid cannot raise or resign.
- Our auction ensures that the bids should be approved by the auction mechanism.

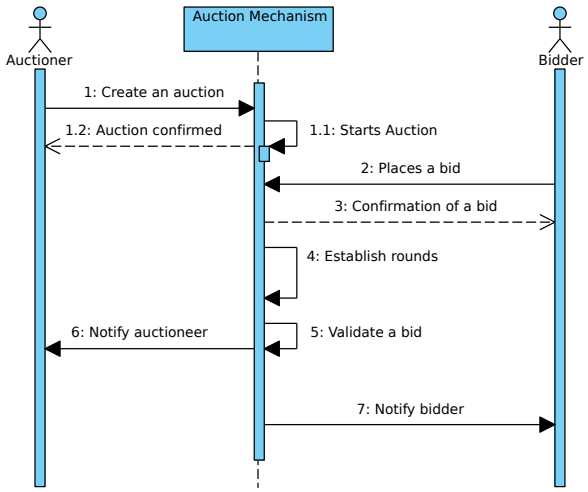

Fig. 1. The process in an auction model based on resource allocation

So as to observe the effect its operational efficiency for convergence of the costs of a resource to a value in a virtual software market, we propose a simulation of our auction based market mechanism, which is implemented in *Mathematica*.

4 A Demonstration of Our Approach

In this section, we illustrate a concrete example of how our approach is applied to a software development process. To simulate auction based market approach, we use the Monte Carlo technique. Consequently, we create a simulation model based on several hypothetical auctions, bidders and auctioneers that are interacting in a virtual (kanban) development process, and generate random variables and related events.

Kanban is a production planning approach that uses a pull system to manage the workflow. There are reasons to choose a kanban scheduling over other software processes. First, it enables us to define "*the software development process in terms of queues and control loops, and manage accordingly*" [30]. Therefore, we find it easy to integrate an market mechanism with kanban workflow. Secondly, as the task allocation process should be continuous in our settings, it is important for us to limit the work in process to the task winners, who should be able to pull the tasks on demand. Thirdly, both a market mechanism and kanban promote the idea of task transparency. Together, they can be used for effective management of the information flow in the software development process.

For our preliminary run, we start with 20 bidders (e.g. software developers), 400 tasks (e.g. coding a unit/function) and 10 auctioneers (e.g. stakeholders). For simplicity, we only assign developer role for all bidders. As soon as the auctions are created, they appear in the system demand pool, and further bidders start the bidding process. By executing the auction rules, in our model 400 tasks were auctioned to the participants. The kanban system identifies the necessary tasks for development, where the virtual market defines the auctioned items. In our approach, we make our calculations based on 120 virtual days of work for 400 auctions, all of which are integrated with the simulation of a virtual kanban workflow.

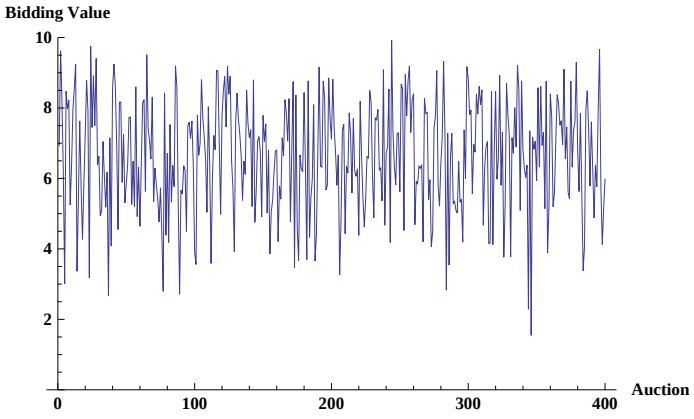

Fig. 2. Task Winners with Bidding Values in 400 Virtual Auctions

Figure 2 provides the distribution of tasks by using the winning bids for the tasks with respect to the number of virtual auctions. It also demonstrates resource consumption levels coupled with the tasks in our market control system. Here, it is important to note that we use the range $[0-10]$ for the bidding values of tasks. In this scenario, the ability of market to mediate the resources in reply to business implications is not as significant as expected, particularly for the planned worked time schedule.

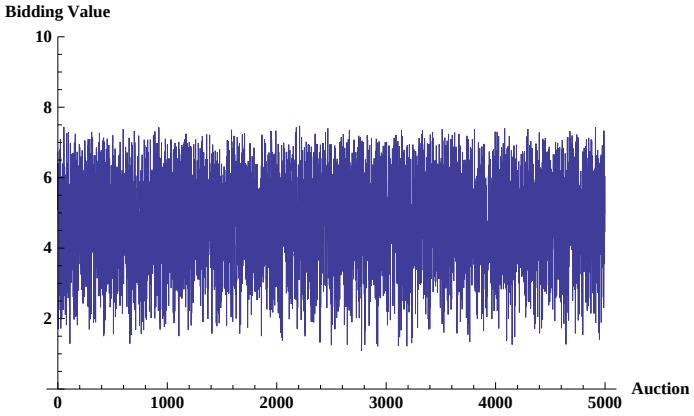

Fig. 3. Task Winners with Bidding Values in 5000 Virtual Auctions

In our second run, we create 5000 auctions for 300 tasks with 100 auctioneers, and 300 bidders. This iteration includes three different roles: (i) software developer, (ii) tester, and (iii) analyst, where we randomly generate different software teams limited to eight participants (up to 500 total). Therefore, in this scenario, our bidders are mostly considered as software teams. Figure 3 illustrates the results of the auctions versus task allocations with winning bids for a second price auction. In addition, it can be inferred from the figure that the health of our market economy relies on the ability of its development process which coordinates the flow of the bidding distribution.

5 Conclusion and future work

As the complexity of software development expands, an increasing amount of resources are required and consequently consumed during a software development process. One way to cope with this problem is through understanding the task allocations achieved during a software development life-cycle. For example, in the field of grid computing, there are a number of studies that propose models for allocating the resources scaled with resource consumers versus their providers in terms of the consumers needs (e.g. [31]). However, task based the resource

allocation problem is not directly considered in a production model from a software management viewpoint, and further in light of a software methodology (i.e. kanban) to create a supply-demand chain that is simulated in an socio-economic landscape of software development.

This work proposes a mechanism for allocation of resources based the resource constrained task problems, particularly in cases either resource allocation is not possible or should be performed dynamically for economic reasons. Such a model can be useful to allocate software development tasks efficiently without the need of a (human) mediator. Based on the Monte Carlo method, we perform two different runs so as to assess the risk analysis on our two different virtual settings. The second iteration (i.e. a large scale run) demonstrates significantly better results.

Although we run a kanban process to control workflow and concurrent development of the distinctive features of a product. On the small scale, however, stabilization of workflow with respect to the auctioned tasks takes more time than expected. However, on a virtually large scale software development landscape - based on our kanban development process - the model confirms that it can be considered as a reasonable strategy for efficient allocation of resources based on our preliminary findings. From a social perspective of a software development process, first model does not allow participants to work together. However, second model enables participants to act cooperatively as a team. In our second hypothetical scenario, evidence suggests that a market based approach will act better than a static allocation technique. In addition, our method also shows that a kanban development process is useful to manage the auction mechanism.

From an industrial perspective, process simulation is an important asset for evaluating alternative scenarios. It is hard to observe which model or scenario works better than the others without simulating the process. We model an auction based market mechanism embedded in a kanban process to simulate virtual participants, auctions, and hypothetical events. By using the *Monte Carlo* method, we simulate our auction based market model with changing bidder roles that affects their behavior, and analyze the outcomes they produce. We argue that auctions processes can be utilized to improve the software development process, where bidder communications (collusion) can be organized to manage software process tasks and activities. In other words, our approach could be found as a convenient method to observe the interacting participants in terms of an auction based market mechanism, on a continuous scale, especially at a large scale software development settings.

Taken together, these results suggest that there is a common ground between auctions and software process improvement concerns, both of which are the processes that are dealing with the optimization of resource constrained task allocations. Finally, we conclude that there are still opportunities for resolving the issues of task allocation problems of the software development processes. In light of this, our proposed mechanism is expected to initiate new directions with its implication in the software process improvement community.

Acknowledgments

This work is supported, in part, by Science Foundation Ireland grant number 03/CE2/I303-1 to Lero, the Irish Software Engineering Research Centre (www.lero.ie).

References

1. Selby, R.: Software engineering: Barry W. Boehm's lifetime contributions to software development, management, and research. Wiley-IEEE Computer Society Pr (2007)
2. Yilmaz, M., OConnor, R., Collins, J.: Improving software development process through economic mechanism design. In: Proceedings of the 17th European Systems and Software Process Improvement and Innovation (EuroSPI 2010). Volume 99., Springer Berlin Heidelberg (2010) 177–188
3. Dittrich, Y., Floyd, C., Klischewski, R.: Social thinking-software practice. The MIT Press (2002)
4. Xiong, J.: New Software Engineering Paradigm Based on Complexity Science: An Introduction to NSE. Springer (2011)
5. Ryan, S., O'Connor, R.V.: Development of a team measure for tacit knowledge in software development teams. Journal of Systems and Software **82** (2009) 229–240
6. Stellman, A., Greene, J.: Applied software project management. O'Reilly Media (2005)
7. Nisan, N.: Algorithmic game theory. Cambridge Univ Pr (2007)
8. Lagesse, B.: A Game-Theoretical model for task assignment in project management. In: 2006 IEEE International Conference on Management of Innovation and Technology, Singapore (2006) 678–680
9. Grechanik, M., Perry, D.E.: Analyzing software development as a noncooperative game. In: IEE Seminar Digests. Volume 29. (2004)
10. Cockburn, A.: The end of software engineering and the start of economic-cooperative gaming. COMSIS **1** (2004) 1–32
11. Cockburn, A.: Agile software development: the cooperative game. Addison-Wesley (2007)
12. Baskerville, R.L., Levine, L., Ramesh, B., Pries-Heje, J.: The high speed balancing game: How software companies cope with internet speed. Scandinavian Journal of Information Systems **16** (2004) 11–54
13. Ko, S.P., Sung, H.K., Lee, K.W.: Study to secure reliability of measurement data through application of game theory. In: Proceedings of the 30th EUROMICRO Conference, Washington, DC, USA, IEEE Computer Society (2004) 380–386
14. Holeman, R.: The software process improvement game. Software Engineering Education (1995) 259–261
15. Ogland, P.: The game of software process improvement: Some reflections on players, strategies and payoff. Norsk konferanse for organisasjoners bruk av informasjonsteknologi (NOKOBIT-16) (2009) 209–223
16. Boehm, B., Ross, R.: Theory-W software project management principles and examples. Software Engineering, IEEE Transactions on **15** (1989) 902–916
17. Sullivan, K., Chalasani, P., Jha, S.: Software design decisions as real options. IEEE Transactions on Software Engineering (1997)
18. Vajja, K.K., TV, P.: Quality attribute game: a game theory based techniquefor software architecture design. In: Proceeding of the 2nd annual conference on India software engineering conference, Pune, India, ACM (2009) 133–134

19. Sazawal, V., Sudan, N.: Modeling software evolution with game theory. Trustworthy Software Development Processes **5543** (2009) 354–365
20. Bavota, G., Oliveto, R., De Lucia, A., Antoniol, G., Gueheneuc, Y.: Playing with refactoring: Identifying extract class opportunities through game theory. In: Software Maintenance (ICSM), 2010 IEEE International Conference on, (IEEE) 1–5
21. Hazzan, O., Dubinsky, Y.: Social perspective of software development methods: The case of the prisoner dilemma and extreme programming. In: Extreme Programming and Agile Processes in Software Engineering. Springer (2005) 74–81
22. Feijs, L.: Prisoner dilemma in software testing. Computer Science Reports **1** (2001) 65–80
23. Oza, N.V.: Game theory perspectives on client: vendor relationships in offshore software outsourcing. (2006) 49–54
24. Klein, M., Moreno, G., Parkes, D., Wallnau, K.: Designing for incentives: better information sharing for better software engineering. In: Proceedings of the FSE/SDP workshop on Future of software engineering research. FoSER '10, ACM (2010) 195–200
25. Hurwicz, L., Reiter, S.: Designing economic mechanisms. Cambridge Univ. Pr. (2006)
26. Zhao, X., Fang, F., Whinston, A.: An economic mechanism for better internet security. Decision Support Systems **45** (2008) 811–821
27. Stef-Praun, T., Rego, V.: Ws-auction: Mechanism design for aweb services market. In: Distributed Computing Systems Workshops, 2006. ICDCS Workshops 2006. 26th IEEE International Conference on, IEEE (2006) 41–41
28. Friedman, E., Parkes, D.: Pricing wifi at starbucks: issues in online mechanism design. In: Proceedings of the 4th ACM conference on Electronic commerce, ACM (2003) 240–241
29. Varian, H.: Computational economics and finance: modeling and analysis with Mathematica. Volume 1. Telos Pr (1996)
30. Shalloway, A., Beaver, G., Trott, J.: Lean-agile software development: achieving enterprise agility. Addison-Wesley Professional (2009)
31. Izakian, H., Abraham, A., Ladani, B.: An auction method for resource allocation in computational grids. Future Generation Computer Systems **26** (2010) 228–235